

# 16. Data Model

Created: September 14, 2004

Updated: September 24, 2004

## Overview

The overview for this chapter consists of the following topics:

- Introduction
- Chapter Outline

## Introduction

The Data Model used in the NCBI sequence database and software tools is described in the C Toolkit documentation. This chapter provides links to the descriptions of particular modules and types. Most of this information is related to ASN.1 specifications. While reading it you can skip parts devoted to the C implementation of the Data Model. This chapter also describes some implementation details specific to the C++ Toolkit.

Another good source of information about the NCBI data model can be found in "Bioinformatics" book:

Bioinformatics  
A Practical Guide to the Analysis of Genes and Proteins  
Second Edition (2001)  
Edited by Andreas D. Baxevanis, B. F. Francis Ouellette  
ISBN 0-471-38391-0

Chapter 2 - The NCBI Data Model

## Chapter Outline

The following is an outline of the topics presented in this chapter:

- Data Model
- General Use Objects
- Bibliographic References
- MEDLINE Data
- Biological Sequences
- Collection of Sequences
- Sequence Locations and Identifiers

- Sequence Features
- Sequence Alignments
- Sequence Graphs

## Data Model

---

The Data Model section outlines the NCBI model for biotechnology information, which is centered on the concept of a biological sequence as a simple, linear coordinate system.

## General Use Objects

---

The General Use Objects section describes the data objects defined in general.asn. They are a miscellaneous collection of generally useful types.

### Module Types

- Large text blocks - StringStore
- Date
- Identifying things - Object-id
- Identifying things - Dbtag
- Identifying people - Person-id
- Expressing uncertainty - Int-fuzz
- Creating your own objects - User-object

## Bibliographic References

---

The Bibliographic References section documents types for storing publications of any sort and collections of publications. The types are defined in biblio.asn and pub.asn modules.

### Module Types

- Citation components:
  - Affil
  - Author-list
  - Imprint
  - Title
- Citing an article - Cit-art
- Citing a journal - Cit-jour
- Citing a book - Cit-book
- Citing proceedings - Cit-proc
- Citing a letter, manuscript, or thesis - Cit-let
- Citing directly submitted data - Cit-sub
- Citing a patent - Cit-pat
- Identifying a patent - Id-pat
- Special cases: unpublished, unparsed, or unusual - Cit-gen
- Accommodating any publication type - Pub
- Grouping different forms of citation for a single work - Pub-equiv
- Sets of citations - Pub-set

## MEDLINE Data

---

The "MEDLINE" section is an introduction to MEDLINE and the structure of a MEDLINE record. It describes types defined in the medline.asn module.

### Module Types

- Structure of a MEDLINE Entry - Medline-entry
- MeSH Index Terms - Medline-mesh

- Substance Records - Medline-rn
- Database Cross Reference Records - Medline-si

## Biological Sequences

---

The Biological Sequences section describes types used to represent biological data. These types are defined in the seq.asn, seqblock.asn, and seqcode.asn modules.

### Module Types

- Biological sequence - Bioseq
- Annotating the Bioseq - Seq-annot
- Describing the Bioseq and placing it in context - Seq-descr
- Instantiating the Bioseq - Seq-inst
- History of a Seq-inst - Seq-hist
- Encoding the sequence data itself - Seq-data
- Tables of sequence codes
- Mapping between different sequence alphabets - Seq-map-table
- Publication describing a Bioseq - Pubdesc
- Applying a numbering system to a Bioseq - Numbering

### C++ Implementation Notes

In the C++ Toolkit some new methods are defined in the classes generated from the ASN.1 specifications. These classes and methods are listed below. Many utility functions for working with Bioseqs and sequence data are defined in the CSeqportUtil class.

- CBioseq:
  - **CBioseq(CSeq\_loc, string)** - constructs a new delta sequence from the Seq-loc. The string argument may be used to specify local Seq-id text for the new Bioseq.

- **GetParentEntry** - returns Seq-entry containing the Bioseq.
- **GetLabel** - returns the Bioseq label.
- **GetFirstId** - returns the first element from the Bioseq's Id list or null.
- **IsNa** - true if the Bioseq is a nucleotide.
- **IsAa** - true if the Bioseq is a protein.
- CSeq\_annot:
  - **AddName** - adds or replaces annotation descriptor of type *name*.
  - **AddTitle, SetTitle** - adds or replaces annotation descriptor of type *title*.
  - **AddComment** - adds annotation descriptor of type *comment*.
  - **SetCreateDate, SetUpdateDate** - add or set annotation create/update time.
  - **AddUserObject** - add a user-object descriptor.
- CSeq\_data - adds constructors to create Seq-data objects from a string or a vector of chars.
- CSeq\_inst - defines **IsNa/IsAa** methods to check sequence type.
- CSeqdesc - defines **GetLabel** method.

## Collection of Sequences

---

The Collection of Sequences section describes the types used to organize multiple Bioseqs into tree structures. The types are located in the seqset.asn module.

### Module Types

- Seq-entry
- Bioseq-set

### C++ Implementation Notes

The C++ Toolkit adds **GetLabel** methods in both CBioseq\_set and CSeq\_entry classes. The **CSeq\_entry** class also defines several methods for accessing the Seq-entry tree structure: **Parentize**, **GetParentEntry**, etc.

## Sequence Locations and Identifiers

---

The Sequence Locations and Identifiers section contains documentation for types used to identify Bioseqs and describe locations on them. These types are defined in the seqloc.asn module.

### Module Types

- Identifying sequences - Seq-id
- Seq-id subtypes
- Location on a Bioseq - Seq-loc

### C++ Implementation Notes

In the C++ Toolkit, many utility functions and classes are added to the module to simplify usage of Seq-id and Seq-loc objects.

CSeq\_loc and some of the subtypes (Seq-interval, Seq-loc-mix etc.) add the following methods to their base classes:

- Constructors to simplify creation of simple Seq-loc objects.
- **GetTotalRange** - returns range, covering the whole Seq-loc. If the Seq-loc refers multiple Bioseqs, exception is thrown.
- **IsReverseStrand** - returns true if all ranges in the Seq-loc have reverse strand.
- **GetStart**, **GetEnd** - return start and stop positions of the Seq-loc. This may be different from **GetTotalRange** if the related Bioseq is circular or if the order of ranges in the Seq-loc is non-standard.
- **GetCircularLength** - returns length of the Seq-loc. If the sequence length is provided, the method checks whether the Seq-loc is circular and calculates the correct length, even if the location crosses a sequence start.
- **CheckId** - checks whether the Seq-loc refers to only one Seq-id and returns it; otherwise, it sends an exception.
- **Compare** - compares two Seq-locs if they are defined on the same bioseq.

- **Add** - adds a sub-location to the existing one.

Beside these methods, a new class **CSeq\_loc\_CI** is defined in Seq\_loc.hpp, which provides simplified access to individual ranges of any Seq-loc, regardless of its real type and structure.

CSeq\_id adds the following methods to its base class:

- Constructors to simplify creation of Seq-ids from primitive types (string, int). Some of these constructors auto-detect the type of the Seq-id from its string representation.
- **IdentifyAccession** - deduces Seq-id information from a bare accession.
- **Match, Compare** - compare Seq-ids.
- **GetTextseq\_Id** - checks whether the Seq-id subtype is Textseq-id compatible and returns its value.
- Several methods for serializing Seq-id or getting its string representation.
- **GetSeq\_idByType, FindGi, FindTextseq\_id** - nonmember template functions to find Seq-id of a particular type in a container.

**IsForward, IsReverse, SameOrientation, Reverse** functions (Na\_strand.hpp) provide the ability to compare or manipulate Na-strand values.

## Sequence Features

---

The Sequence Features section documents data structures used to describe regions of Bioseqs. The types are located in the seqfeat.asn module.

### Module Types

- Structure of a feature - Seq-feat
- Type-specific feature data - SeqFeatData
- Coding region - Cdregion
- Genetic-code
- Reference to a restriction enzyme - Rsite-ref
- Reference to an RNA - RNA-ref
- Reference to a gene - Gene-ref

- Reference to a protein - Prot-ref
- Transcription initiation - Txinit

#### C++ Implementation Notes

In the C++ Toolkit, many types defined in the seqfeat ASN.1 module are extended to simplify access to the feature data. The CSeq\_feat class has methods for comparing features by type and location. The CSeqFeatData class defines feature subtypes and qualifiers so that you can better identify individual features.

## Sequence Alignments

The Sequence Alignments section is devoted to the data structures used to describe relationships (mappings) between several Bioseqs. The types are located in the seqalign.asn module.

#### Module Types

- Seq-align
- Score of an alignment or segment - Score
- Segments for "diags" Seq-align - Dense-diag
- Segments for "global" or "partial" Seq-align - Dense-seg
- Aligning any Bioseq type with any other - Std-seg

#### C++ Implementation Notes

The C++ Toolkit adds several methods to the classes generated from ASN.1 specifications to simplify alignment data access and manipulation. The CSeq\_align class has methods returning Sqe-id, start, stop, and strand for a particular alignment row, regardless of its representation; it allows swapping alignment rows or converting the alignment from one type to another. The CDense\_seg class extends the default set of alignment members with sequence character width (1 or 3, depending on molecule type).

## Sequence Graphs

---

The Sequence Graphs section describes Seq-graph type used to associate some analytical data with a region on a Bioseq. The type definition is located in the seqres.asn module.